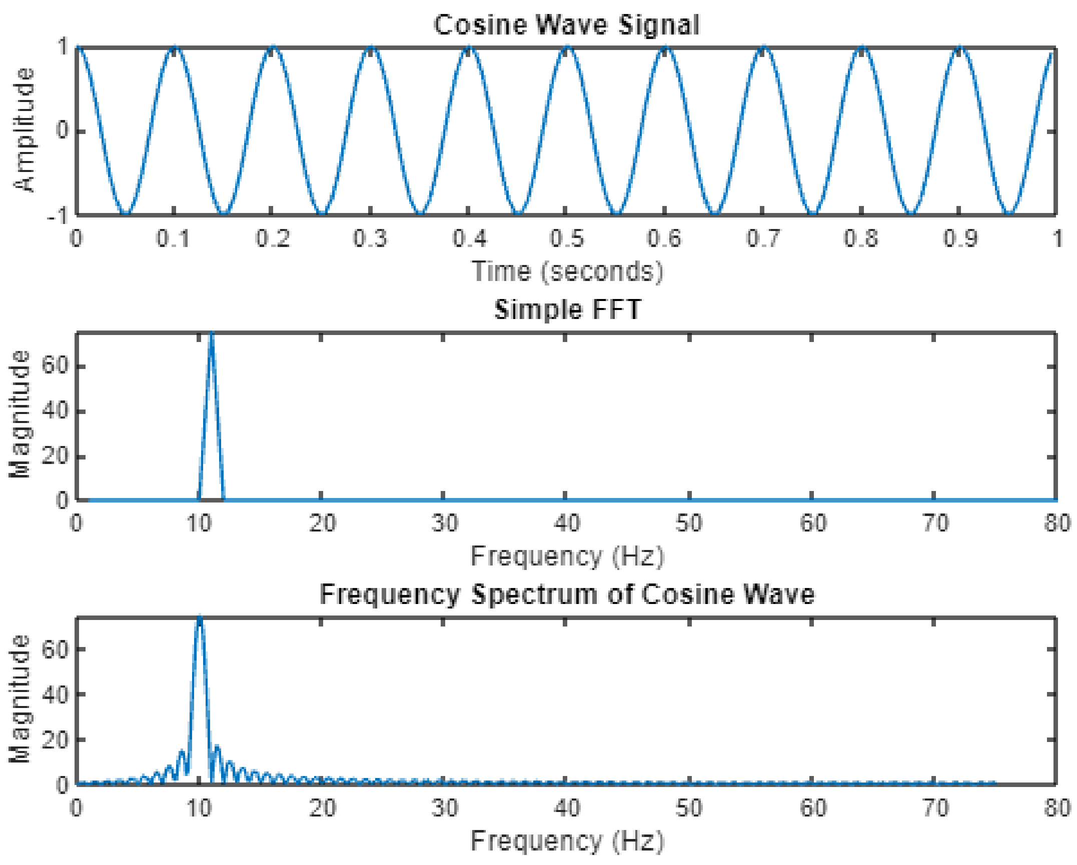
Task 01:

A-Frequency Analysis of a cosine signal:

- Generate a cosine wave of frequency 10 Hz.
- Take the fft of cosine wave generated in part (b).
- Take the N point fft of cosine wave generated in part (c).
- Draw time domain and frequency domain signals in one plot.

```
% Define sampling frequency (Fs) in Hz
Fs = 150;  % Choose a sampling frequency higher than the signal frequency

% Define frequency of the cosine wave (f) in Hz
f = 10;

% Create time vector (t) for 1 second with samples based on Fs
t = 0:1/Fs:1-1/Fs;

% Generate the cosine wave signal (x)
x = cos(2*pi*f*t);

y = fft(x);

% Define length of the FFT (nfft)
nfft = 1024;  % Choose an FFT length that is a power of 2 for efficiency

% --- Part (b): Take the FFT of cosine wave ---
% Compute the Fast Fourier Transform (FFT) of x with padding to nfft length
X = fft(x, nfft);

% Extract the positive frequency components and magnitude (absolute values)
X_pos = X(1:floor(nfft/2));
mag_X = abs(X_pos);

% Create frequency vector (f_pos) based on Fs and nfft for positive frequencies
f_pos = linspace(0, Fs/2, (nfft/2));

% --- Part (c): Take the N point FFT of cosine wave ---
% Use the same approach as in part (b) but with a different nfft value
% --- Plt time domain and frequency domain signals ---
figure(1);
% Plot the cosine wave signal in the time domain
subplot(3,1,1);
plot(t, x);
title('Cosine Wave Signal');
xlabel('Time (seconds)');
ylabel('Amplitude');
subplot(3,1,2);
plot(real(y));
xlim([0 80])
title('Simple FFT');
```

```
xlabel('Frequency (Hz)');
ylabel('Magnitude');
% Combine plots for positive frequencies from both FFTs in the frequency domain
subplot(3,1,3);
plot(f_pos, mag_X);
title('Frequency Spectrum of Cosine Wave');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```



Explanation:

In this code, we're exploring the properties of a cosine wave using a technique called the Fast Fourier Transform (FFT). First, we set the sampling frequency (Fs) to 150 Hz, which is higher than the frequency of the cosine wave (f), set at 10 Hz. Then, we create a time vector (t) for 1 second with samples based on Fs. Using this time vector, we generate the cosine wave signal (x). The FFT of this signal is computed, and we visualize it in the frequency domain. We also take a specific number of points (nfft = 1024) to improve efficiency. Part (b) of the code calculates the FFT, extracts positive frequency components, and plots the frequency spectrum. In part (c), you're encouraged to experiment with a different nfft value. The three subplots in the figure depict the time domain representation of the cosine wave, a simple FFT plot, and the frequency spectrum of the cosine wave. The code is a hands-on exploration of signal processing concepts, helping you understand how to analyze signals in both time and frequency domains.

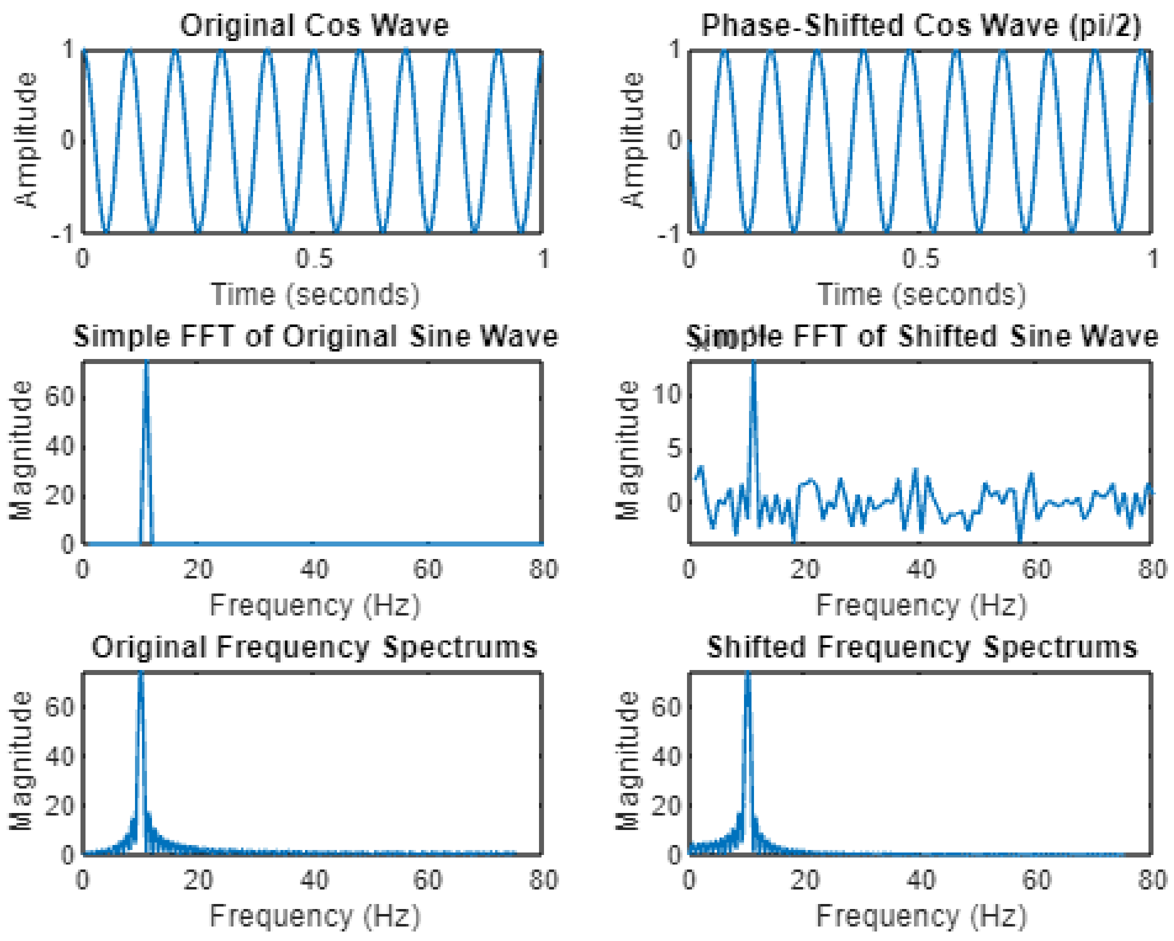
Task 02:

B-Frequency Analysis of a shifted sine wave:

- Run the code in Task # 1.
- Insert a phase shift of pi/2 in sine wave generated in part (a).
- Take the fft of phase shifted sine wave.
- Take the N point fft of shifted cosine wave.
- Explain the difference you noticed in original and shifted wave's time and frequency domain plot.

```
% Define sampling frequency (Fs) in Hz
Fs = 150;  % Choose a sampling frequency higher than the signal frequency

% Define frequency of the sine wave (f) in Hz
f = 10;

% Create time vector (t) for 1 second with samples based on Fs
t = 0:1/Fs:1-1/Fs;

% Generate the original sine wave signal (x)
x_original = cos(2*pi*f*t);

% Generate the phase-shifted sine wave signal (x_shifted) with pi/2 shift
x_shifted = cos(2*pi*f*t + pi/2);

% Take the FFT of the original sine wave (y_original)
y_original = fft(x_original);

y_shifted = fft(x_shifted);
% Define length of the FFT (nfft)
nfft = 1024;  % Choose an FFT length that is a power of 2 for efficiency


X = fft(x_original, nfft);

% Extract the positive frequency components and magnitude (absolute values)
X_pos = X(1:floor(nfft/2));
mag_X = abs(X_pos);


% --- Part (b): Take the FFT of phase-shifted sine wave ---
% Compute the Fast Fourier Transform (FFT) of x_shifted with padding to nfft length
X_shifted = fft(x_shifted, nfft);

% Extract the positive frequency components and magnitude (absolute values)
X_pos_shifted = X_shifted(1:floor(nfft/2));
mag_X_shifted = abs(X_pos_shifted);

% Create frequency vector (f_pos) based on Fs and nfft for positive frequencies
f_pos = linspace(0, Fs/2, (nfft/2));
```

```
% --- Plot time domain and frequency domain signals ---
figure(1);

% Plot the original sine wave signal in the time domain
subplot(3,2,1);
plot(t, x_original);
title('Original Cos Wave');
xlabel('Time (seconds)');
ylabel('Amplitude');

% Plot the phase-shifted sine wave signal in the time domain
subplot(3,2,2);
plot(t, x_shifted);
title('Phase-Shifted Cos Wave (pi/2)');
xlabel('Time (seconds)');
ylabel('Amplitude');

% Simple FFT of the original sine wave (for reference)
subplot(3,2,3);
plot(real(y_original));
xlim([0 80])
title('Simple FFT of Original Sine Wave');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Combine plots for positive frequencies from FFT of shifted sine wave
subplot(3,2,4);
plot(real(y_shifted));
xlim([0 80])
title('Simple FFT of Shifted Sine Wave');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Additional subplot to compare original and shifted frequency spectrums (optional)
subplot(3,2,5);
%plot(t, x_original);
plot(f_pos, mag_X);  % Full spectrum of original sine wave
title('Original Frequency Spectrums');
xlabel('Frequency (Hz)');
ylabel('Magnitude');

% Additional subplot to compare original and shifted frequency spectrums (optional)
subplot(3,2,6);
%plot(t, x_original);
plot(f_pos, mag_X_shifted);
title('Shifted Frequency Spectrums');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```



Explanation:

In this code, we're examining the effects of phase shifting on a sine wave using the Fast Fourier Transform (FFT). First, we set the sampling frequency (Fs) to 150 Hz and define the frequency of the original sine wave (f) as 10 Hz. Two sine waves are generated: the original sine wave (x_original) and a phase-shifted version (x_shifted) with a pi/2 (90 degrees) phase shift. The FFT is applied to both signals, and positive frequency components are extracted and plotted in the frequency domain. The subplots in the figure display the time domain representation of the original and phase-shifted sine waves, along with the FFT plots for reference. The last two subplots compare the frequency spectrums of the original and phase-shifted signals. This code allows you to visually understand how phase shifting impacts the frequency components of a sine wave.

Task 03:

C-Frequency Analysis of a square wave:

- Generate a square wave of any frequency without square command. (Hint: Add odd harmonics of sin wave)
- Take the fft of square wave generated in part (a).

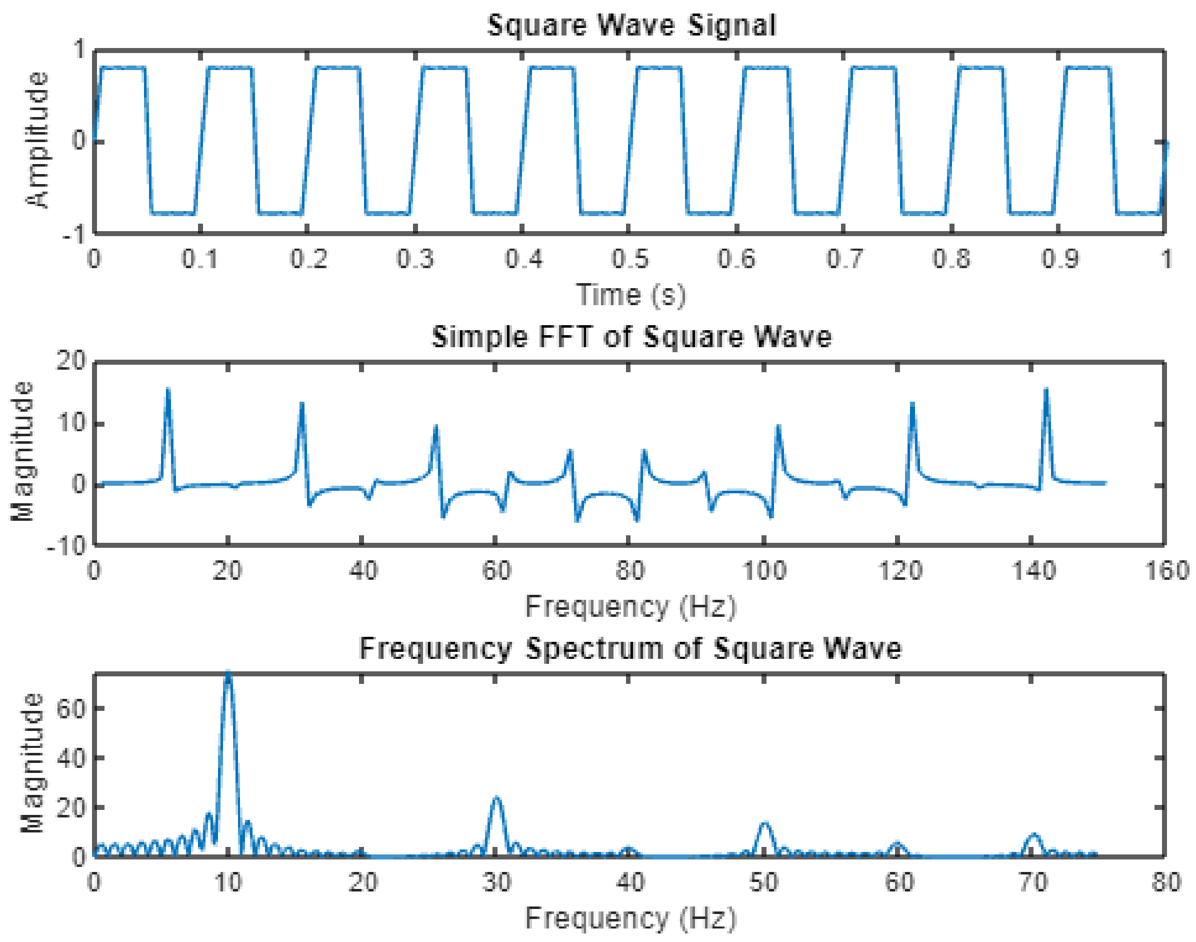
- Draw time domain and frequency domain signals.
- Also explain the frequency plot of square wave

```
clc, clear all
Fs = 150; % Sampling frequency
t = 0:1/Fs:1; % Time vector of 1 second

% Define fundamental frequency (f) of the square wave
f = 10;

x = sin(2*pi*t*f);

% Add higher order odd harmonics (3rd, 5th, ...)
a = input('Enter number of harmonics: ')
```

```
a = 40
```

```
b = 1 + 2*a;
for n = 3:2:b % Loop through odd harmonics up to 29th
  harmonic_freq = f * n;
  % Add harmonic with smaller amplitude as n increases
  x = x + (1/(n)) * sin(2*pi*t*harmonic_freq);
end

nfft = 1024; % Length of FFT


y = fft(x)
plot(real(y))
% Take FFT, padding with zeros
X = fft(x, nfft);

% Extract single-sided magnitude
X = X(1:nfft/2);
mag_x = abs(X);
% Frequency vector
f = (0:nfft/2-1)*Fs/nfft;
% Plot time domain signal
figure(1);
subplot(3,1,1)
plot(t, x);
title('Square Wave Signal');
xlabel('Time (s)');
ylabel('Amplitude');
subplot(3,1,2)
plot(real(y))
title('Simple FFT of Square Wave');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
% Plot frequency domain (magnitude spectrum)
```

```
subplot(3,1,3)
plot(f, mag_x);
title('Frequency Spectrum of Square Wave');
xlabel('Frequency (Hz)');
ylabel('Magnitude');
```



Explanation:

In this code, we're generating and analyzing a square wave signal with added higher-order odd harmonics. The sampling frequency (Fs) is set to 150 Hz, and the fundamental frequency (f) of the square wave is chosen as 10 Hz. Initially, a basic sine wave is created. The user is prompted to input the number of harmonics to add, and then a loop generates and adds odd harmonics (starting from the 3rd harmonic up to the specified limit). As the loop progresses, each harmonic is added with a decreasing amplitude. The code then performs the FFT on the resulting signal, padding zeros for efficiency. The time domain signal, the simple FFT plot, and the frequency spectrum are visualized in three subplots. This code helps you explore how a square wave can be constructed by combining multiple harmonics and provides insights into the resulting frequency spectrum.

Conclusion:

In this lab, we delved into the fascinating world of signal processing and Fourier analysis. Beginning with a cosine wave, we explored the Fast Fourier Transform (FFT) to examine its frequency components. Subsequently, we extended our understanding to phase-shifting a sine wave and scrutinized the impact on its frequency spectrum. Lastly, we tackled the generation of a square wave enriched with higher-order odd harmonics, unraveling the intricacies of its frequency composition. Through visualizations in both time and frequency domains, this lab provided valuable hands-on experience, enhancing our comprehension of signals, harmonics, and their transformations.